Lab wk1-2: Recursive generation of Combinatorial Objects

Generating the Power Set of a set recursively

Goal: practice your recursive programming skills and prepare for the next assignment.

Given a set A, the set of all its subsets is called its Power set, and is usually denoted $\mathcal{P}(A)$. The **number** of subsets of a finite set = $|P(A)|$ is = $2^{|A|}$.

For example: if A = {a, b, c} then $\mathcal{P}(A)$ has 8 elements (written $|\mathcal{P}(A)| = 8$) since |A| = 3 and the number of subsets is 2^3. The subsets are: {}, {a}, {b}, {c}, {a,b}, {a,c}, {b,c}, {a,b,c}

Write a recursive Java method that will generate all the subsets of the letters in a string (which is passed as an explicit parameter) and return the subsets as an ArrayList of strings. You must follow the high-level pseudo-code given below.

```
getSubsets(setString : a string with the characters that make up the set)
     let A and temp be empty ArrayLists
     if len(setString)>0
          temp = getSubsets (string without last character)
          //  now loop over temp and create the subsets with and without
          //  the last character of the original string
          for (int i = 0; i < temp.size(); i++)
               A.add(temp.get(i))  // adds subsets without last character
          for (int i = 0; i < temp.size(); i++)
               A.add(temp.get(i) + last character of a)
                         //adds subsets with last character
          return A
     else           // the empty set is the only subset of the empty set
          A.add("")   // array list with only the empty string
          return A
```

Before you implement this make sure you can draw the call tree if it is called on "abc" so you are sure you understand what is going on.

A template for your program and a simple driver program is provided on PolyLearn
Source code for a single class, **SubsetGen.java** with the method described below. Submit on PolyLearn.

public class SubsetGen
contains the method
public ArrayList<String> getSubsets (String word) {}
When you getSubsets method is called with the word = "abc", it should return the following 8 strings:

"empty string" a b ab c ac bc abc

It also contains the method, getGrayCode(int n), described in the second part of this lab.

The bitstring representation of sets. Given a set A containing n-elements. Associate each elements with a number from 1 to n=|A|, then any subset can be represented by a bitstring of length n. E.g.
A = {a,b,c,d} then the subset {a,c} is represented by 1010 if we have associate a with position 1, b with position 2 etc. This is a bijection between P(A) the set of all subsets of A with the set of all the bitstrings of length |A| Subsets are examples of combinatorial objects.

This is an efficient way to represent subsets by using a 1 or 0 in the i-th position to represent that either the i-th item is in the subset or it is not in the subset. For some applications it is most efficient if a subset differs from the previous subset in at most one position. That is, either a single 0 will change to a 1 or a single 1 will change to a 0 with all the other bits remaining the same. This is called a Gray code.

You are to implement the following recursive method in the class SubsetGen:

- **getGrayCode(int n)** that returns an ArrayList of strings where each string represents a bitstring (contains only the characters 0 and 1) that represent the subsets of a set containing n elements. Your method **must return the Gray Code as described below**.
- Your program must be well structured, commented, and easy to read.
- the method must be **recursive and must follow the high level description below or it may not pass the tests**. See below for the desired output for the gray code of sets with 2 or 3 elements.

1. Take all the subsets of the n-1 items in a Gray code order and prepend a 0 to create a subset of n items without the n-th item.
2. Then reverse the same list of subsets of the n-1 items in a Gray code order and prepend a 1 to them to create subsets of items that do contain the n-th item,

Implement this **recursively** with a base case of an ArrayList of with two strings "0" "1" representing the two subset of a set with a single item.
A call to your function for sets with 2 items would return "00" "01" "11" "10".

For 3 items it would be "000" "001" "011" "010" "110" "111" "101" "100"
Note the first four strings above are the four strings returned for a two item set but with "0" prepended and then the last four are the four strings returned for two item set in the reverse order with "1" prepended.

Make sure to test it for 4 items. Manually generate the Gray code for four items as described above and have one of the TA's check that you have done it correctly.